SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO–MATEMATIČKI FAKULTET
MATEMATIČKI ODSJEK

Petra Međimurec

ALGORITMI ELEMENTARNE TEORIJE BROJEVA I NEKE NJIHOVE PRIMJENE

Diplomski rad

Voditelj rada:
izv. prof. dr. sc. Zrinka Franušić

Zagreb, 2019.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Ovaj rad posvećujem svojim bližnjima bez kojih moje postignuće ne bi bilo moguće. Ujedno zahvaljujem zaručniku, roditeljima, prijateljima i kolegama na pruženoj potpori i razumijevanju tijekom cijelog studija. Posebna zahvala mentorici izv. prof. dr. sc. Zrinki Franušić što ste našli vremena za moja pitanje te mi svojim savjetima pomogli u izradi rada.

Diplomski rad napravljen je u sklopu aktivnosti Projekta KK.01.1.1.01.0004 - Znanstveni centar izvrsnosti za kvantne i kompleksne sustave te reprezentacije Liejevih algebri.

Sadržaj

Uvod

Algoritam je metoda, proces, postupak, tehnika ili pravilo za rješavanje neke klase problema. Algoritam se kao postupak za dobivanje rješenja sastoji od konačnog niza koraka, koji se trebaju izvršiti da bi se dobilo rješenje. Neki primjeri algoritama su pripremanje jela uz recept, upute za korištenje nekog aparata, postupak za množenje dva prirodna broja, Euklidov algoritam za traženje zajedničke mjere dva prirodna broja, itd.

Probleme rješavamo s ciljem da dobijemo rješenje, ali u razumnom vremenu. Složenost algoritma je cijena korištenja (izvođenja) tog algoritma za rješavanje jednog problema iz određene klase. Mjeri se u vremenu izvršavanja, potrebnoj memoriji ili sličnim relevantnim pojmovima. Kod složenosti algoritama razmatra se vremenska složenost i memorijska složenost. Vremenska složenost je vrijeme potrebno da se algoritam izvrši. Memorijska složenost je memorija koja je potrebna da se algoritam izvrši.

U ovom radu ćemo prikazati osnovne algoritme iz elementarne teorije brojeva i neke njihove primjene. Jedan od najstarijih algoritama u neprestanoj uporabi je *Euklidov algoritam*. To je vrlo učinkovita procedura za izračunavanje najvećeg zajedničkog djelitelja dva broja. Nosi ime po starogrčkom matematičaru Euklidu, koji ga je prvi put opisao u svojim Elementima (oko 300. godine prije Krista). Euklidov algoritam ima veliku primjenu, neke od primjena ćemo i prikazati. Posebno koristan pokazuje se Prošireni Euklidov algoritam koji za dva prirodna broje a i b prikazuje njihov najveći zajednički djelitelj kao cjelobrojnu linearnu kombinaciju, $ax + by = d$. Taj se algoritam može implementirati u rješavanju linearnih kongruencija i sustava linearnih kongruencija.

Razvoj broja u verižni razlomak također ima algoritamsku strukturu i nalazi primjenu u rješavanju nekih diofantskih jednadžbi. Na kraju predstavljamo i nešto od praktične primjene teorije brojeva, primjene u kriptografiji. Algoritamski opisujemo jedan od najpoznatiji kriptosustava s javnim ključem, tzv. RSA kriptosustav. Njegova sigurnost se zasniva na teškoći faktorizacije velikih složenih brojeva oblika

$n = pq$, gdje su p, q prosti. Pod pojmom *veliki* mislimo na brojeve od stotinjak znamenaka. Nije lako pronaći tako velike proste brojeve. Stoga možemo reći da je traženje velikih prostih brojeva, osim ljudskom radoznalošću, potaknuto i potrebama iz kriptografije. Za ispitivanje prostosti prikazat ćemo dva jednostavnija vjerojatnosna testa - Fermatov i Millerov.

Algoritmi iz ovog rada su prezentirani u pseudo-jeziku te je njihov kod dan u programskom jeziku *Python 3.6.*

Poglavlje 1

Euklidov algoritam

1.1 Osnovni Euklidov algoritam

Definicija 1.1. *Neka su $a \neq 0$ i b cijeli brojevi. Kažemo da je b* ***djeljiv*** *s a, odnosno da a* ***dijeli*** *b, ako postoji cijeli broj x takav da je $b = ax$. To zapisujemo sa $a \mid b$. Ako b nije djeljiv s a onda pišemo $a \nmid b$.*

Definicija 1.2. *Neka su b i c cijeli brojevi, cijeli broj a zovemo* ***zajednički djelitelj*** *od b i c ako je $a \mid b$ i $a \mid c$. Ako je barem jedan od brojeva b i c različit od nule, onda postoji konačno mnogo djelitelja od b i c. Najveći među njima zove se* ***najveći zajednički djelitelj*** *od b i c te se označava s $\gcd(b, c)$.*

Analogno se može definirati najveći zajednički djelitelj konačno mnogo cijelih brojeva $b_1, b_2, \ldots, b_n$ od kojih je bar jedan različit od nule. Njega označavamo s $\gcd(b_1, b_2, \ldots, b_n)$.

Teorem 1.3 (Teorem o dijeljenju s ostatkom)**.** *Za proizvoljan prirodan broj a i cijeli broj b postoje jedinstveni brojevi q i r takvi da je $b = qa + r$ i $0 \leq r < a$.*

Dokaz. Promotrimo skup $\{b - am : n \in \mathbb{Z}\}$. Najmanji nenegativni član ovog skupa označimo sa r. Tada je po definiciji $0 \leq r < a$ i postoji $q \in \mathbb{Z}$ takav da je $b - qa = r$, tj. $b = qa + r$. Da bi dokazali jedinstvenost od q i r, pretpostavimo da postoji još jedan par q_1, r_1 koji zadovoljava iste uvjete, $b = q_1a + r_1$ i $0 \leq r_1 < a_1$. Oduzimanjem dviju jednakosti za b dobivamo $0 = (q_1 - q)a + r_1 - r$. Pretpostavimo da je npr. $r < r_1$. Tada je $0 \leq r_1 - r < a$, odnosno $0 \leq a(q - q_1) < a$. Kako je $a \in \mathbb{N}$ i $q - q_1 \in \mathbb{Z}$, prethodna nejednakost je jedino moguća za $q = q_1$. Prema tome je i $r_1 = r$. □

Teorem 1.4. *Neka su $a, b \in \mathbb{Z}$. Tada vrijedi*

$$\gcd(a, b) = \min\left(\{ax + by : x, y \in \mathbb{Z}\} \cap \mathbb{N}\right).$$

Dokaz. Neka je $d = \gcd(a, b)$, te neka je l najmanji pozitivni član skupa $S = \{ax + by : x, y \in \mathbb{Z}\}$. To znači da postoje cijeli brojevi x_0, y_0 takvi da je $l = ax_0 + by_0$. Pokažimo da je $l \mid a$ i $l \mid b$. Pretpostavimo da $l \nmid a$. Tada po Teoremu 1.3 postoje cijeli brojevi q i r takvi da je $a = lq + r$ i $0 < r < l$. Sada je $r = a - lq = a - q(ax_0 + by_0) = b(1 - qx_0) + c(-qy_0) \in S$, što je u suprotnosti s minimalnošću od l. Dakle, $l \mid a$. Na isti način pokazali bismo da je $l \mid b$. Stoga, $l \mid g$. S druge strane je očito da $g \mid ax_0 + by_0 = l$. Budući da su g i l prirodni brojevi te $l \mid g$ i $g \mid l$ slijedi $g = l$, što je i trebalo pokazati. □

Propozicija 1.5. $\gcd(a, b) = \gcd(a, b + ax)$

Dokaz. Označimo $\gcd(a, b) = d$, $\gcd(a, b + ax) = g$. Prema Teoremu 1.4 postoje $x_0, y_0 \in \mathbb{Z}$ takvi da je $d = ax_0 + by_0$, odnosno $d = a(x_0 - xy_0) + (b + ax)y_0$. Odavde slijedi da je $g|d$. Pokažimo sada da je $d|g$. Budući da $d|a$ i $d|b$ te imamo da $d|(b+ax)$. Dakle, d je zajednički djelitelj od a i $b + ax$, te slijedi da je $d|g$. □

Teorem 1.6 (Euklidov algoritam)**.** *Neka su a i b prirodni brojevi. Pretpostavimo da je uzastopnom primjenom teorema o dijeljenju s ostatkom dobiven niz jednakosti*

$$\begin{array}{rl} a = bq_1 + r_1, & 0 < r_1 < b, \\ b = r_1q_2 + r_2, & 0 < r_2 < r_1, \\ r_1 = r_2q_3 + r_3, & 0 < r_3 < r_2, \\ \vdots & \\ r_{j-2} = r_{j-1}q_j + r_j, & 0 < r_j < r_{j-1}, \\ r_{j-1} = r_jq_{j+1}. & \end{array} \tag{1.1}$$

Tada je $\gcd(a, b) = r_j$.

Dokaz. Po Propoziciji 1.5 imamo: $\gcd(a, b) = \gcd(a - bq_1, b) = \gcd(r_1, b) = \gcd(r_1, b - r_1q_1) = \gcd(r_1, r_2) = \gcd(r_1 - r_2q_3, r_2) = \gcd(r_3, r_2)$. Ako nastavimo, dobivamo: $\gcd(a, b) = \gcd(r_{j-1}, r_j) = \gcd(r_j, 0) = r_j$. □

Iz prethodnog dokaza vidimo da se primjena Euklidovog algoritam za traženje najveće zajedničke mjere zasniva na Propoziciji 1.5, odnosno na jednakosti

$$\gcd(a, b) = \gcd(b,\ a \mod b),$$

pri čemu je $a \mod b$ ostatak pri dijeljenju broja a brojem b, tj. r_1.

Napomenimo još da se u (1.1) često uvodi oznaka $a = r_{-1}$, $b = r_0$.

Primjer 1.7. *Odredimo najveći zajednički djelitelj od* 180 *i* 146.

Rješenje.

$$\begin{aligned}\gcd(180,146) &= \gcd(146, 180 \mod 146) = \gcd(146,34)\\ &= \gcd(34, 146 \mod 34) = \gcd(34,10)\\ &= \gcd(10, 34 \mod 10) = \gcd(10,4)\\ &= \gcd(4, 10 \mod 4) = \gcd(4,2)\\ &= \gcd(2, 4 \mod 2) = \gcd(2,0) = 2\end{aligned}$$

Algoritam 1. *Euklidov algoritam*

`Ulaz:` $a, b \in \mathbb{N}$
`Sve dok` $b \neq 0$ `radi`
$p = a \bmod b$, $a = b$, $b = p$
`Izlaz:` a

Iterativni Euklidov algoritam u programskom jeziku Python:

```
def gcd(a,b):
       while b!=0:
            p = a % b
            a = b
            b = p
       return a
```

Rekurzivni Euklidov algoritam u programskom jeziku Python:

```
def gcd(a,b):
     if b==0:
           return a
     else:
           return gcd(b, a % b)
```

Propozicija 1.8. *Broj koraka u Euklidovom algoritmu* (1.1) *je manji od* $2log_2 b$.

Dokaz. U i-tom, $1 \leq i < j$ koraku algoritma (1.1) računa se ostatak r_i za koji vrijedi $0 < r_i < r_{i-1}$. Može nastupiti točno jedan od slučajeva:

(i) $r_i \leq \frac{r_{i-1}}{2}$,

(ii) $\frac{r_{i-1}}{2} < r_i < r_{i-1}$.

Pokažimo da oba slučaja povlače nejednakost $r_{i+1} \leq \frac{r_{i-2}}{2}$, za sve $i = 1, \ldots, j-1$ (uz $a = r_{-1}$, $b = r_0$).

(i) Za ostatak r_{i+1} dobiven u koraku $i+1$ vrijedi $r_{i+1} < r_i$. Stoga je $r_{i+1} \leq \frac{r_{i-2}}{2}$.

(ii) U ovom slučaju, u sljedećem, odnosno $(i+1)$-om koraku, dobivamo da je

$$q_{i+1} = 1.$$

Stoga je $r_{i+1} = r_{i-1} - r_i < r_{i-1} - \frac{r_{i-1}}{2} = \frac{r_{i-1}}{2}$.

Ako je broj koraka j u algoritmu (1.1) paran tada dobivamo

$$1 \leq r_j < \frac{r_{j-2}}{2} < \frac{r_{j-4}}{4} < \ldots < \frac{r_0}{2^{j/2}} = \frac{b}{2^{j/2}},$$

a u slučaju kada je j neparan vrijedi

$$2 \leqslant r_{j-1} < \frac{r_{j-3}}{2} < \ldots < \frac{r_0}{2^{(j-1)/2}} = \frac{b}{2^{(j-1)/2}}.$$

Stoga je

$$2^{j/2} < b.$$

Logaritmiranjem prethodne nejednakosti dobivamo traženu gornju ogradu. □

Primjer 1.9. *Broj koraka Euklidovog algoritma primijenjenog na dva susjedna Fibonaccijeva broja, F_{n+1} i F_n je n.*

Rješenje. Fibonaccijevi brojevi definirani su rekurzijom $F_{n+1}=F_n + F_{n-1}$, gdje je $F_0 = 0, F_1 = F_2 = 1$. Tražimo $\gcd(F_{n+1}, F_n)$. Tada je $a = F_{n+1}$ i $b = F_n$. Primjenom Euklidovog algoritma dobivamo:

$$\begin{array}{rcll} F_{n+1} & = & F_n + F_{n-1}, & 0 < F_{n-1} < F_n, \\ F_n & = & F_{n-1} + F_{n-2}, & 0 < F_{n-2} < F_{n-1}, \\ F_{n-1} & = & F_{n-2} + F_{n-3}, & 0 < F_{n-3} < F_{n-2}, \\ \vdots & & & \\ F_3 & = & F_2 + F_1, & 0 < F_1 < F_2 \\ F_2 & = & F_1 + F_0 & \end{array}$$

Iz $F_0 = 0$, slijedi da je $\gcd(F_{n+1}, F_n) = F_1 = 1$. Broj koraka Euklidovog algoritma je n.

1.2 Prošireni Euklidov algoritam

U Teoremu 1.4 je pokazano da se najveći zajednički djelitelj brojeva a i b može prikazati kao cjelobrojna linearna kombinacija od a i b, tj. postoje $x_0, y_0 \in \mathbb{Z}$ takvi da je

$$\gcd(a, b) = x_0 a + y_0 b.$$

Prošireni Euklidov algoritam, uz $\gcd(a, b)$, računa i cjelobrojne koeficijente x_0 i y_0.

Propozicija 1.10. *Svaki ostatak r_i, $1 \leq i \leq j$, iz (1.1) je cjelobrojna linearna kombinacija od a i b.*

Dokaz. Tvrdnju dokazujemo primjenom principa matematičke indukcije.

Baza: r_1 i r_2 su cjelobrojne linearne kombinacije od a i b. Zaista, $r_1 = a + (-q_1)b$ i $r_2 = b - q_2 r_1 = (-q_2)a + (1 + q_1 q_2)b$.

Pretpostavka indukcije: Pretpostavimo da su r_{i-1} i r_{i-2} cjelobrojne linearne kombinacije od a i b.

Korak: Budući da je r_i cjelobrojna linearna kombinacija od r_{i-1} i r_{i-2}, lakim računom i primjenom pretpostavke indukcije dobivamo da je i r_i cjelobrojna linearna kombinacija od a i b. □

Prikažimo niz jednakosti iz algoritma (1.1) matrično:

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} q_1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} b \\ r_1 \end{bmatrix},$$

$$\begin{bmatrix} b \\ r_1 \end{bmatrix} = \begin{bmatrix} q_2 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \end{bmatrix},$$

$$\begin{bmatrix} r_1 \\ r_2 \end{bmatrix} = \begin{bmatrix} q_3 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_2 \\ r_3 \end{bmatrix},$$

$$\vdots$$

$$\begin{bmatrix} r_{j-2} \\ r_{j-1} \end{bmatrix} = \begin{bmatrix} q_j & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_{j-1} \\ r_j \end{bmatrix},$$

$$\begin{bmatrix} r_{j-1} \\ r_j \end{bmatrix} = \begin{bmatrix} q_{j+1} & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} d \\ 0 \end{bmatrix},$$

gdje je $d = r_j = \gcd(a, b)$ i $r_{j+1} = 0$. Uvrštavanjem unazad dolazimo do jednakosti:

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} q_1 & 1 \\ 1 & 0 \end{bmatrix} \cdots \begin{bmatrix} q_{j+1} & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} d \\ 0 \end{bmatrix}.$$

Definiramo matricu M_j

$$M_j = \begin{bmatrix} A_j & B_j \\ C_j & D_j \end{bmatrix} = \begin{bmatrix} q_1 & 1 \\ 1 & 0 \end{bmatrix} \cdots \begin{bmatrix} q_{j+1} & 1 \\ 1 & 0 \end{bmatrix}.$$

Primjenom Binetovog teorema jednostavno možemo izračunati determinantu matrice M_j

$$\det M_j = \det \begin{bmatrix} q_1 & 1 \\ 1 & 0 \end{bmatrix} \cdots \det \begin{bmatrix} q_{j+1} & 1 \\ 1 & 0 \end{bmatrix} = (-1)^{j+1}$$

pa zaključujemo da je M_j regularna matrica. Njezin inverz se lako može odrediti prema formuli:

$$M_j^{-1} = (-1)^{j+1} \begin{bmatrix} D_j & -B_j \\ -C_j & A_j \end{bmatrix}.$$

Iz jednakosti

$$\begin{bmatrix} a \\ b \end{bmatrix} = M_j \begin{bmatrix} d \\ 0 \end{bmatrix},$$

slijedi da je

$$\begin{bmatrix} d \\ 0 \end{bmatrix} = M_j^{-1} \begin{bmatrix} a \\ b \end{bmatrix} = (-1)^{j+1} \begin{bmatrix} D_j & -B_j \\ -C_j & A_j \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}.$$

Otuda je

$$d = \underbrace{(-1)^{j+1} D_j}_{=x_0} a + \underbrace{(-1)^j B_j}_{=y_0} b.$$

Dakle, traženi koeficijenti su $x_0 = (-1)^{j+1} D_j$ i $y_0 = (-1)^j B_j$.

Primjer 1.11. *Za* $a = 180, b = 146$ *izračunajmo* $\gcd(a, b)$*,* x_0 *i* y_0 *pomoću matrice* M_j*.*

Rješenje. Primjenom Euklidovog algoritma redom dobivamo:

$$q_1 = 1, q_2 = 4, q_3 = 3, q_4 = 2, q_5 = 2, d = 2$$

Sada je

$$M_4 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 4 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 90 & 37 \\ 73 & 30 \end{bmatrix},$$

pa slijedi da je $x_0 = -30$, $y_0 = 37$.

Algoritam 2. *Prošireni Euklidov algoritam I*

`Ulaz:` $a, b \in \mathbb{N}$
$M := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
$j := 0$
`Sve dok` $b \neq 0$ `radi`

$$q = \lfloor a/b \rfloor,\ M = M \cdot \begin{bmatrix} q & 1 \\ 1 & 0 \end{bmatrix},\ p = a - bq,\ a = b, b = p, j = j + 1$$

`Izlaz:` $d = a$, $x_0 = (-1)^j M[2,2]$, $y_0 = (-1)^{j+1} M[1,2]$

Algoritam u programskom jeziku Python. (U algoritam je implementirana i funkcija `umnozak` za množenje matrica).

```
C = [[0,0],
     [0,0]]

def umnozak(A,B,C):
        C = [[0,0],
             [0,0]]
        for i in range(len(A)):
          for j in range(len(B[0])):
             for k in range(len(B)):
                C[i][j] += A[i][k] * B[k][j]
        return C

def prosireni_euklid(a, b):
            M = [[1,0],[0,1]]
            j = 0
            if a<b:
               p=a
               a=b
               b=p
            while b!=0:
               q = a//b
               B = [[q,1],[1,0]]
               M = umnozak(M,B,C)
```

```
        p = a-q*b
        a = b
        b = p
        j = j+1
d = a
x = ((-1)**j)*M[1][1]
y = ((-1)**(j+1))*M[0][1]
print('x =',x,'y =',y,'d =',d)
```

Rješenja jednadžbe $ax+by = \gcd(a,b)$ možemo "iščitati" i iz osnovnog Euklidovog algoritma kao što se može vidjeti na sljedećem primjeru.

Primjer 1.12. *Odredimo* $\gcd(153,21)$ *i prikažimo* $\gcd(153,21)$ *kao linearnu kombinaciju od* 153 *i* 21 *pomoću osnovnog Euklidovog algoritma.*

Rješenje.

$$153 = 21 \cdot 7 + 6$$
$$21 = 6 \cdot 3 + 3$$
$$6 = 3 \cdot 2$$

Dakle, $\gcd(153,21) = 3$. Nadalje imamo:

$$3 = 21 - 6 \cdot 3 = 21 - (153 - 21 \cdot 7) \cdot 3 = 8 \cdot 21 - 153 \cdot 3.$$

Metoda prikazana u Primjeru 1.12 nije baš praktična. Zbog toga ćemo definirati rekurzivne nizove (x_i) i (y_i) koji zadovoljavaju istu rekurzivnu relaciju kao i niz ostataka (r_i) te početne uvjete pomoću kojih ćemo odrediti rješenja jednadžbe $ax + by = \gcd(a,b)$. Prema (1.1) vrijedi:

$$r_{-1} = a, \quad r_0 = b, \quad r_i = r_{i-2} - q_i r_{i-1},$$

za $i = 1, \ldots, j$. Definiramo:

$$x_{-1} = 1, \quad x_0 = 0, \quad x_i = x_{i-2} - q_i x_{i-1},$$
$$y_{-1} = 0, \quad y_0 = 1, \quad y_i = y_{i-2} - q_i y_{i-1},$$

za $i = 1, \ldots, j$.

Propozicija 1.13. *Vrijedi*

$$ax_i + by_i = r_i,$$

za $i = -1, 0, 1, 2, \ldots, j+1$. *Specijalno,* $ax_j + by_j = \gcd(a,b)$.

Dokaz. Matematičkom indukcijom, slično kao u dokazu Propozicije 1.10. □

Algoritam 3. *Prošireni Euklidov algoritam II*

`Ulaz:` $a, b \in \mathbb{N}$
$(x_0, x_1, y_0, y_1) := (0, 1, 1, 0)$
`Sve dok` $b \neq 0$ `radi`
$p = a \bmod b$, $q = (a - p)/b$, $a = b$, $b = p$
$p = x_1$, $x_1 = x_0 - q \cdot x_1$, $x_0 = p$
$p = y_1$, $y_1 = y_0 - q \cdot y_1$, $y_0 = p$
`Izlaz:` a, x_0, y_0

```
def prosireni_gcd(a, b):
          x0, x1, y0, y1 = 0, 1, 1, 0
          while a != 0:
               q, b, a = b // a, a, b % a
               y0, y1 = y1, y0 - q * y1
               x0, x1 = x1, x0 - q * x1
          return b, x0, y0
```

Analogno iterativnom algoritmu radi i rekurzivni algoritam. U rekurzivnom algoritmu, varijable zamjenjujemo u pozivu definirane funkcije te u povratnoj vrijednosti algoritam računa ono što u prethodnom algoritmu računa u svakom koraku iteracije:

```
def rek_gcd(a, b):
      if a == 0:
           return (b, 0, 1)
      else:
           q, x, y = rek_gcd(b % a, a)
      return (q, y - (b // a) * x, x)
```

Složenost proširenog Euklidovog algoritma je $O(ln(a)ln(b))$. Prošireni Euklidov algoritam posebno je koristan kada je $\gcd(a, b) = 1$. Koristi se za računanje multiplikativnog inverza, koji se koristi RSA metodi šifriranja javnog ključa, koju ćemo u nastavku spomenuti.

Primjer 1.14. *Napiši algoritam koji za* $a, b \in [5, 10]$ *ispisuje sve vrijednosti* x, y *za koje vrijedi* $ax + by = \gcd(a, b)$.

Rješenje.

Kod programa:

```
def rek_gcd(a, b):          #funkcija za prošireni Euklidov algoritam
           if a == 0:
                  return (b, 0, 1)
           else:
                  q, x, y = rek_gcd(b % a, a)
                  return (q, y - (b // a) * x, x)

for i in range(5, 10, 1):
      for j in range(10, 5, -1):
           a = i
           b = j
           print('a =',i,'b =',j)
           d, x, y = rek_gcd(i,j)
           print(a, '*', x, '+', b, '*', y, '=', d)
```

Primjer ispisa:

```
a = 5 b = 10
5 * 1 + 10 * 0 = 5
a = 5 b = 9
5 * 2 + 9 * -1 = 1
a = 5 b = 8
5 * -3 + 8 * 2 = 1
a = 5 b = 7
5 * 3 + 7 * -2 = 1
a = 5 b = 6
5 * -1 + 6 * 1 = 1
```

Poglavlje 2

Linearne kongruencije

2.1 Kongruencije

Definicija 2.1. *Ako cijeli broj $m \neq 0$ dijeli razliku $a - b$, onda kažemo da je a* ***kongruentan*** *b* ***modulo*** *m i pišemo $a \equiv b \pmod{m}$. U protivnom, kažemo da a* ***nije kongruentan*** *b* ***modulo*** *m i pišemo $a \not\equiv b \pmod{m}$.*

Propozicija 2.2. *Neka su a, b, c, d cijeli brojevi. Ako je $a \equiv b \pmod{m}$ i $c \equiv d \pmod{m}$, onda vrijedi:*

1. $a \pm c \equiv b \pm d \pmod{m}$,
2. $ac \equiv bd \pmod{m}$,
3. $ac \equiv bc \pmod{m}$, *za* $c \neq 0$.

Dokaz. Postoje $k, l \in \mathbb{Z}$ tako da je $a - b = mk$ i $c - d = ml$. Zbrajanjem tih jednakosti dobivamo da je $(a + c) + (b + d) = m(k + l)$ te slijedi da je $a + c \equiv b + d \pmod{m}$. Oduzimanjem tih jednakosti dobivamo $(a-c)-(b-d) = m(k-l)$ te slijedi $a - c \equiv b - d \pmod{m}$. Množenjem jenakosti $a = b + mk$ i $c = d + ml$ dobivamo $ac = bd + m(kd + lb + mkl)$ iz čega slijedi da $m \mid ac - bd$, odnosno tvrdnja 3. Iz $a - b = mk$ slijedi $ac - bc = (mc)k$, pa je $ac \equiv bc \pmod{m}$. □

Propozicija 2.3. *Ako je $a \equiv b \pmod{m}$, tada je $a^n \equiv b^n \pmod{m}$, $n \in \mathbb{N}$.*

Dokaz. Tvrdnju dokazujemo primjenom principa matematičke indukcije.

Baza: $n = 1$, slijedi $a \equiv b \pmod{m}$.

Pretpostavka indukcije: Pretpostavimo da $a^n \equiv b^n \pmod{m}$ za neki $n \in \mathbb{N}$.

Korak: Prema tvrdnji 3. iz Propozicije 2.2 vrijedi $a \cdot a^n \equiv b \cdot b^n \pmod{m}$, odnosno $a^{n+1} \equiv b^{n+1} \pmod{m}$. □

Teorem 2.4. *Vrijedi* $ax \equiv ay \pmod{m}$ *ako i samo ako* $x \equiv y \pmod{\frac{m}{\gcd(a,m)}}$. *Posebno, ako je* $ax \equiv ay \pmod{m}$ *i* $\gcd(a,m) = 1$, *onda je* $x \equiv y \pmod{m}$.

Dokaz. Postoji $k \in Z$ takav da je $ay - ax = mk$. Tada je $\frac{a}{\gcd(a,m)}(y-x) = \frac{m}{\gcd(a,m)}$, slijedi $\frac{m}{\gcd(a,m)} \mid \frac{a}{\gcd(a,m)}(y-x)$. No brojevi $\frac{a}{\gcd(a,m)}$ i $\frac{m}{\gcd(a,m)}$ su relativno prosti, pa zaključujemo da $\frac{m}{\gcd(a,m)} \mid y-x$ te $x \equiv y \pmod{m}$.

Obrnuto, ako je $x \equiv y \pmod{\frac{m}{\gcd(a,m)}}$, onda po Propoziciji 2.2 dobivamo $ax \equiv ay \pmod{\frac{am}{\gcd(a,m)}}$. Pošto je $\gcd(a,m)$ djelitelj od a, po Propoziciji 2.2 dobivamo da je $ax \equiv ay \pmod{m}$. □

2.2 Rješavanje linearne kongruencije

Neka su $a, b \in \mathbb{Z}$, $m \in \mathbb{N}$. Riješiti linearnu kongruenciju

$$ax \equiv b \pmod{m}, \tag{2.1}$$

znači odrediti sve cijele brojeve x koji ju zadovoljavaju. Očito, ako je $x_0 \in \mathbb{Z}$ neko rješenje od (2.1) onda će to biti i $x_0 + km$ za svaki $k \in \mathbb{Z}$. Takva rješenja, odnosno međusobno kongruentna rješenja modulo m, nazivamo *ekvivalentnima*. Zato ćemo se ograničiti na određivanje rješenja od (2.1) u skupu $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$. Napomenimo da uz operacije zbrajanja modulo m i množenja modulo m, skup $\mathbb{Z}_m$ ima strukturu komutativnog prstena s jedinicom.

Ako je m dovoljno mali sva rješenja od (2.1) mogu se naći i provjerom za svaki element iz $\mathbb{Z}_m$ redom. No, ovdje ćemo prikazati i znatno učinkovitiji algoritam koji koristi Euklidov algoritam. Za početak treba nešto reći o rješivosti i broju rješenja kongruencije (2.1).

Teorem 2.5. *Neka su* $a, b \in \mathbb{Z}$ *i* $m \in \mathbb{N}$. *Kongruencija* (2.1) *ima rješenja ako i samo ako* $\gcd(a,m)$ *dijeli* b. *Ako je kongruencija* (2.1) *rješiva, onda u prstenu* $\mathbb{Z}_m$ *ih ima točno* $\gcd(a,m)$ *rješenja.*

Dokaz. Ako kongruencija $ax \equiv b \pmod{m}$ ima rješenja, onda postoji $y \in \mathbb{Z}$ tako da je $ax - my = b$. Slijedi da $\gcd(a,m) \mid b$.

Pretpostavimo da $d = \gcd(a,m)$ dijeli b. Stavimo $a' = \frac{a}{d}, b' = \frac{b}{d}, m' = \frac{m}{d}$. Trebamo riješiti kongruenciju $a'x \equiv b' \pmod{m'}$. Kongruencija ima točno jedno rješenje. Budući da je $\gcd(a', m') = 1$ i ako je $x \in \mathbb{Z}_{m'}$, slijedi $a'x \in \mathbb{Z}_{m'}$, to jest svaki ostatak modulo m' (tako i b') se dobiva točno za jedan $x \in \mathbb{Z}_{m'}$. Ako je x' neko rješenje od $a'x \equiv b' \pmod{m'}$ onda su sva rješenja od $ax \equiv b \pmod{m}$ dana s $x = x' + nm'$, gdje je $n = 0, 1, ..., d-1$. Dakle, ako d dijeli b, onda kongruencija $ax \equiv b \pmod{m}$ ima točno d rješenja modulo m.

□

Promotrimo, najprije, specijalnu kongruenciju

$$ax \equiv 1 \pmod{m}. \tag{2.2}$$

Prema Teoremu 2.5, kongruencija (2.2) je rješiva ako i samo ako je $\gcd(a, m) = 1$ te u tom slučaju ima jedinstveno rješenje u $\mathbb{Z}_m$. To rješenje naziva se *modularni inverz* od a. Zapravo, to rješenje je jedinstveni *multiplikativni inverz* od $a_0 \in \mathbb{Z}_m$ u prstenu $\mathbb{Z}_m$, gdje je $a_0 \equiv a \pmod{m}$. Označavat ćemo ga s a^{-1}.

Sljedeći algoritam traži modularni inverz modulo m broja a redom u nizu $1, 2 \ldots, m$. Ukoliko "dosegne" m , zaključujemo da a nije invertibilan.

Algoritam 4. *Modularni inverz I*

`Ulaz:` $a, m \in \mathbb{Z}$
$i := 1$
`Sve dok` $(a \mod m) \neq 1$ `i` $i < m$ `radi`
    $a = i * a$
    $i = i + 1$
`Ako` $i = m$ `onda`
    `Izlaz:` $\gcd(a, b) > 1$
`Inače`
    `Izlaz:` i

Algoritam u programskom jeziku Python:

```
def mod_inverz(a, m):
        i = 1
        while (a * i) % m != 1 and i<m:
                i +=1
        if i == m:
               print('gcd(a,b)>1')
        else:
               return i
```

Algoritam se sastoji od petlje koja se u najgorem slučaju ponavlja m puta te za svaki prirodan broj $x < m$ provjerava vrijedi li svojstvo modularnog inverza. Složenost algoritma je $O(n)$.

Ako je $\gcd(a, m) = 1$, onda prema Teoremu 2.5 kongruencija (2.1) ima jedinstveno rješenje

$$x \equiv a^{-1}b \pmod{m}.$$

Primjer 2.6. *Riješimo linearnu kongruenciju* $73x \equiv 20 \pmod{13}$.

Rješenje. Budući da je $\gcd(73, 13) = 1$, postoji modularni inverz od a, tj. a^{-1}. Koristimo algoritam za računanje modularnog inverza (Algoritam 4):

```
Ulaz: mod_inverz(73,13)
Izlaz: 5.
```

Dakle, rješenje je

$$x \equiv 5 \cdot 20 \equiv 9 \pmod{13}.$$

U nastavku ćemo predstaviti algoritam za rješavanje linearne kongruencije koji se zasniva na Proširenom Euklidovom algoritmu. Ako je $\gcd(a, m) = 1$, tada prema Teoremu 1.4 postoje $x_0, y_0 \in \mathbb{Z}$ takvi da vrijedi $ax_0 + my_0 = 1$. Tada je

$$ax_0 \equiv 1 \pmod{m}, \tag{2.3}$$

pa je

$$a(x_0 b) \equiv b \pmod{m}.$$

Stoga je

$$x \equiv x_0 b \pmod{m}$$

rješenje kongruencije (2.1). Vrlo lako ovu ideju možemo primijeniti i na opći slučaj u kojem je $\gcd(a, m) = d$ i $d \mid b$. Naime, kongruencija (2.1) je prema Teoremu 2.4 ekvivalentna kongruenciji

$$a'x \equiv b' \pmod{m)'},$$

gdje su $a' = \frac{a}{d}$, $b' = \frac{b}{d}$ i $m' = \frac{m}{d}$. Uočimo da je $\gcd(a', m') = 1$ pa dalje dobivamo, analogno,

$$x \equiv x_0 b' \pmod{m)'},$$

pri čemu je $x_0 \in \mathbb{Z}$ takav da vrijedi 2.3, odnosno $a'x_0 \equiv 1 \pmod{m}$. Uz uznaku $x_1 = x_0 b'$, sva rješenja modulo m kongruencije (2.1) su

$$x \equiv x_1, x_1 + m', x_1 + 2m', \ldots, x_1 + (d-1)m' \pmod{m}.$$

Dakle, primijenimo li Prošireni Euklidov algoritam a i m dobit ćemo podatak o broju rješenja modulo m (ako rješenje postoji) te dobiti rješenje kongruencije (2.1) modulo m'.

Algoritam 5. *Rješavanje linearne kongruencije*

`Ulaz:` $a, b, m \in \mathbb{N}$
$(x_0, x_1) := (0, 1)$
`Sve dok` $m \neq 0$ `radi`
$p = a \bmod m$, $q = (a - p)/m$, $a = m$, $m = p$
$p = x_1$, $x_1 = x_0 - q \cdot x_1$, $x_0 = p$
`Ako` $b \bmod a = 0$ `onda`
`Izlaz:` a, $x_0 \cdot b/a$
`inače`
`Izlaz:` `Nema rješenja`

Algoritam u programskom jeziku Python:

```
def linearna_kon(a, b, m):
          x2 = m
          x0 = 1
          x1 = 0
          while m!=0:
              p = a%m
              q = (a-p)//m
              a = m
              m = p
              p = x1
              x1 = x0 -q*x1
              x0 = p
          if b%a==0:
                return a, (x0*b%x2)//a
          else:
                print('Nema rjesenja')
```

Primjer ispisa:

```
Ulaz: linearna_kon(312, 39, 765)
Izlaz: 3, 32
```

```
Ulaz:linearna_kon(15, 10, 75)
Izlaz: Nema rjesenja
```

Primjer 2.7. *Riješimo linearnu kongruenciju* $445x \equiv 615 \pmod{715}$.

Rješenje. Primjenimo Euklidov algoritam:

$$\begin{gathered} 715 = 445 \cdot 1 + 270 \\ 445 = 270 \cdot 1 + 175 \\ 270 = 175 \cdot 1 + 95 \\ 175 = 95 \cdot 1 + 80 \\ 95 = 80 \cdot 1 + 15 \\ 80 = 15 \cdot 5 + 5 \\ 15 = 5 \cdot 3 \end{gathered}$$

Budući da je $\gcd(445, 715) = 5$ i $5 \mid 615$, početna kongruencija je ekvivalentna kongruenciji $89 \equiv 123 \pmod{143}$. Slijedi da je $\gcd(143, 89) = 1$. Pa možemo $\gcd(143, 89)$ prikazati kao linearnu kombinaciju brojeva 143 i 89, $143 \cdot (-28) + 89 \cdot 45 = 1$. Dakle rješenje kongruencije $89x \equiv 1 \pmod{143}$ je $x \equiv 45 \pmod{143}$. Slijedi da je rješenje kongruencije $89x \equiv 123 \pmod{143}$ dano s $x \equiv 45 \cdot 123 \pmod{143} \equiv 5535 \pmod{143} \equiv 101 \pmod{143}$. Budući da je $\gcd(445, 715) = 5$, slijedi da početna linearna kongruencija ima 5 rješenja

$$x \equiv 101, 101 + 143, 101 + 2 \cdot 143, 101 + 3 \cdot 143, 101 + 4 \cdot 143 \pmod{143},$$

$$x \equiv 101, 244, 387, 530, 673 \pmod{143}.$$

Rješenje Primjera 2.7 pomoću Algoritma 5.

```
Ulaz: linearna_kon(445, 615, 715)
Izlaz: (5, 101)
```

2.3 Kineski teorem o ostatcima

Teorem 2.8 (Kineski teorem o ostatcima)**.** *Neka su $m_1, m_2, \ldots, m_r$ u parovima relativno prosti prirodni brojevi, te neka su $a_1, a_2, \ldots, a_r$ cijeli brojevi. Tada sustav kongruencija*

$$x \equiv a_1 \pmod{m_1}, x \equiv a_2 \pmod{m_2}, \ldots, x \equiv a_r \pmod{m_r} \tag{2.4}$$

ima rješenja. Ako je x_0 jedno rješenje, onda su sva rješenja od 2.4 u $\mathbb{Z}$ dana s

$$x \equiv x_0 \pmod{m_1 m_2 \cdots m_r}.$$

Dokaz. Neka je $m = m_1 m_2 \cdots m_r$, te neka je $n_j = \frac{m}{m_j}$ za $j = 1, ..., r$. Tada je $\gcd(m_j, n_j) = 1$, pa postoji cijeli broj x_j takav da je $n_j x_j \equiv a_j \pmod{m_j}$. Za broj

$$x_0 = n_1 x_1 + \cdots + n_r x_r$$

vrijedi da je $x_0 \equiv 0 + \cdots + 0 + n_j x_j + 0 + \cdots + 0 \equiv a_j \pmod{m_j}$ za sve $j = 1, \ldots, r$. Prema tome, x_0 je rješenje sustava (2.4).

Ako su $x, y \in \mathbb{Z}$ dva rješenja od (2.4), onda je $x \equiv y \pmod{m_j}$ za $j = 1, ..., r$, zbog toga jer su m_j u parovima relativno prosti, dobivamo da je $x \equiv y \pmod{m}$. □

Dokaz prethodnog teorema je važan jer nam upravo daje efektivan postupak za određivanje jednog rješenja od (2.4) koji ćemo implementirati u sljedeći algoritam.

Algoritam 6. *Kineski teorem o ostatcima I*

`Ulaz:` $A = [a_1, a_2, ..., a_r], M = [m_1, m_2, ..., m_r]$
$x_0 = 0$
$m = m_1 m_2 \cdots m_r$
`za` $j = 1$ `do` j `radi`
    `odredi` x `takav da je` $(m/m_j)x \equiv a_j \pmod{m_j}$ (Algoritam 5)
    $x_0 = x_0 + x \cdot n_j$
$x_0 = x_0 \mod m$
`Izlaz:` x

Prethodni algoritam rješava jednu po jednu linearnu kongruenciju. Tako da nakon j-tog koraka u petlji x zadovoljava $x \equiv a_j \pmod{m_j}$ za $j = 1, 2, ..., i + 1$. No, rješavanju sustava kongruencija možemo pristupiti na još jedan način. Pretpostavimo da se sustav kongruencija sastoji od dvije linearne kongruencije:

$$x \equiv a_1 \pmod{m_1},\ x \equiv a_2 \pmod{m_2}. \tag{2.5}$$

Primjenom Proširenog Euklidovog algoritma dobivamo $x_1, y_1 \in \mathbb{Z}$ za koje

$$x_1 m_1 + y_1 m_2 = 1.$$

Tada je

$$x = (x_1 m_1 a_2 + y_1 m_2 a_1) \mod m_1 m_2$$

rješenje sustava (2.5). Zaista,

$$x_1 m_1 a_2 + y_1 m_2 a_1 \equiv y_1 m_2 a_1 = (1 - x_1 m_1) a_1 \equiv a_1 \pmod{m_1},$$

te analogno $x_1 m_1 a_2 + y_1 m_2 a_1 \pmod{m_2}$. Nadalje, prema Teoremu 2.8 znamo da sustav (2.5) ima jedinstveno rješenje modulo $m_1 m_2$. Sada ovu ideju možemo općenito primijeniti na sustav kongruencija (2.4). U prvom koraku na opisani način dobili bi rješenje prve kongruencije, $x \equiv a_2' \pmod{m_1 m_2}$ gdje je

$$a_2' = x_0 m_1 a_2 + y_0 m_2 a_1.$$

Zatim bi ponovo rješavali sustav od dvije linearne kongruencije

$$x \equiv a_2' \pmod{m_1 m_2},\ x \equiv a_3 \pmod{m_3},$$

te dobili rješenje

$$x \equiv a_3' \pmod{m_1 m_2 m_3},$$

gdje je

$$a_3' = x_2 (m_1 m_2) a_3 + y_2 m_3 a_2',$$

pri čemu su $x_2, y_2 \in \mathbb{Z}$ takvi da je $x_2(m_1 m_2) + y_2 m_3 = 1$ jer je $\gcd(m_1 m_2, m_3) = 1$, a dobili smo ih pomoću Proširenog Euklidovog algoritma. Nastavljajući postupak dalje, u koraku $r-1$ rješavali bi sustav

$$x \equiv a_{r-1}' \pmod{m_1 m_2 \cdots m_{r-1}},\ x \equiv a_r \pmod{m_r}$$

te bi konačno rješenje glasilo

$$x \equiv x_{r-1}(m_1 m_2 \cdots m_{r-1}) a_r + y_{r-1} m_r a_{r-1}',$$

pri čemu su $x_{r-1}, y_{r-1} \in \mathbb{Z}$ takvi da je $x_{r-1}(m_1 m_2 \cdots m_{r-1}) + y_{r-1} m_r = 1$, jer je $\gcd(m_1 m_2 \cdots m_{r-1}, m_r) = 1$, a a_{r-1}' je rješenje sustava koji se sastoji od prvih $r-1$ kongruencija.

Algoritam 7. *Kineski teorem o ostatcima II*

`Ulaz:` $A = [a_1, a_2, ..., a_r], M = [m_1, m_2, ..., m_r]$
$m = m_1$
$x = a_1$
`for` $(2 \leq i \leq r)$
`nađi` x_0, y_0 `takve da vrijedi` $x_0 m + y_0 m_i = 1$ (Alg. 2 ili 3)
$x = x_0 m a_1 + y_0 m_i x$
$m = m m_i$

$x = x \mod m$

Izlaz: x

Algoritam u programskom jeziku Python:

```
def CRT(A, M):
          m = M[0]
          x = A[0]
          for i in range(1, len(A)):
               d, x0, y0 = rek_gcd(m, M[i])
               x = x0 * m * A[i] + y0 * M[i] * x
               m = m * M[i]
               x = x % m
          return x
```

Primjer ispisa:

```
Ulaz:  A = [3, 5, 6]
       M = [4, 7, 13]
       CRT(A, M)
Izlaz:  19
```

Primjer 2.9. *Riješimo sustav kongruencija*

$$\begin{aligned} x &\equiv 5 \pmod{24}, \\ x &\equiv 29 \pmod{55}, \\ x &\equiv 39 \pmod{50}. \end{aligned}$$

Rješenje. Uočimo da brojevi 24, 55, 50 nisu u parovima relativno prosti, pa ne možemo Kineski teorem o ostacima primijeniti direktno, a moguće je da sustav ni nema rješenja. Vrijedi:

$$\begin{aligned} x \equiv 5 \pmod{24} &\Leftrightarrow x \equiv 5 \pmod{3},\ x \equiv 5 \pmod{8} \\ x \equiv 29 \pmod{55} &\Leftrightarrow x \equiv 29 \pmod{11},\ x \equiv 29 \pmod{5} \\ x \equiv 39 \pmod{50} &\Leftrightarrow x \equiv 39 \pmod{2},\ x \equiv 39 \pmod{25} \end{aligned}$$

Sada na sustav

$$x \equiv 2 \pmod{3},\ x \equiv 5 \pmod{8},\ x \equiv 7 \pmod{11},\ x \equiv 9 \pmod{25}$$

možemo primijeniti Kineski teorem o ostatcima. Koristimo algoritam 7:

```
Ulaz: A= [2, 5, 7, 9], M= [2, 8, 11, 25]
Izlaz: 5309
```

Poglavlje 3

Verižni razlomci

3.1 Razvoj u verižni razlomak

Definicija 3.1. *Neka je $x \in \mathbb{R}$.* ***Najveće cijelo*** *od x, u oznaci $\lfloor x \rfloor$ je najveći cijeli broj koji nije veći od x.*

Neka je $\alpha \in \mathbb{R}$. Definiramo $a_0 = \lfloor \alpha \rfloor$. Ako je $a_0 \neq \alpha$, onda α možemo zapisati u obliku

$$\alpha = a_0 + \frac{1}{\alpha_1},$$

gdje je $\alpha_1 > 1$. Stavimo $a_1 = \lfloor \alpha_1 \rfloor$. Ako je $a_1 \neq \alpha_1$, onda α_1 možemo zapisati u obliku

$$\alpha_1 = a_1 + \frac{1}{\alpha_2},$$

gdje je $\alpha_2 > 1$, te definiramo $a_2 = \lfloor \alpha_2 \rfloor$. Proces nastavljamo dalje ako je $a_2 \neq \alpha_2$. Ako za neki $n \in \mathbb{N}$ dobivamo $a_n \neq \alpha_n$, onda se postupak zaustavlja i broj α ima sljedeći oblik:

$$\alpha = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \ddots + \cfrac{1}{a_n}}} \tag{3.1}$$

Kažemo da je (3.1) prikaz broja α u *konačni verižni razlomak*. Očito je da ako vrijedi (3.1), onda je α racionalan broj. No, vrijedi i obrat. Stoga je $\alpha \in \mathbb{Q}$ ako i samo ako mu je razvoj u verižni razlomak konačan. S druge strane, opisani postupak se ponavlja u beskonačnost ako i samo ako je α iracionalan broj. U tom slučaju

njemu se pridružuje *bekonačni verižni razlomak*. Ako opisani postupak ponavljamo u beskonačnost dobivamo oblik:

$$\alpha = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \ddots + \cfrac{1}{a_{n-2} + \cfrac{1}{a_{n-1}+\alpha_n}}}} \tag{3.2}$$

Verižni razlomak oblika (3.1) kraće zapisujemo kao $[a_0; a_1, a_2, ...a_n]$, a oblika (3.2) kraće zapisujemo kao $[a_0; a_1, \ldots, a_{n-1}, \ldots]$.

Uočimo da je u (3.1) $a_0 \in \mathbb{Z}$, te $a_i \in \mathbb{N}$. Brojevi a_i nazivaju se *parcijalni kvocijenti* verižnog razlomka.

Za početak ćemo odrediti razvoj u verižni razlomak racionalnog broja. Umjesto postupka opisanog na početku koristit ćemo Euklidov algoritam. Lako možemo vidjeti da će kvocijenti dobiveni Euklidovim algoritmom upravo biti parcijalni kvocijenti verižnog razlomka.

Primjer 3.2. *Razvijmo* $\frac{42}{11}$ *u verižni razlomak.*

Rješenje. Primijenimo Euklidov algoritam (1.1):

$$\begin{aligned} 42 &= 11 \cdot 3 + 9, \\ 11 &= 9 \cdot 1 + 2, \\ 9 &= 2 \cdot 4 + 1, \\ 2 &= 1 \cdot 2. \end{aligned}$$

Svaki redak algoritma možemo zapisati na slijedeći način:

$$\begin{aligned} \frac{42}{11} &= 3 + \frac{9}{11}, \\ \frac{11}{9} &= 1 + \frac{2}{9}, \\ \frac{9}{2} &= 4 + \frac{1}{2}. \end{aligned}$$

Stoga je

$$\frac{42}{11} = 3 + \frac{9}{11} = 3 + \frac{1}{\frac{11}{9}} = 3 + \frac{1}{1 + \frac{2}{9}} = 3 + \frac{1}{1 + \frac{1}{\frac{9}{2}}} = 3 + \frac{1}{1 + \frac{1}{4 + \frac{1}{2}}},$$

odnosno kraće $\frac{42}{11} = [3; 1, 4, 2]$.

Ako je $\alpha \in \mathbb{R}$, onda algoritam za razvoj u verižni razlomak dobivamo direktno iz definicije. Budući da se algoritam može beskonačno ponavljati ako je α iracionalan, moramo unijeti i duljinu razvoja n.

Algoritam 8. *Razvoj u verižni razlomak*

`Ulaz:` $x \in \mathbb{R}, n \in \mathbb{N}$
$x_1 = x$
$A[0] = \lfloor x_1 \rfloor$
$i = 1$
`dok` $i < n$ `i` $x_1 \notin \mathbb{Z}$ `radi`
$x_1 = 1/(x_1 - \lfloor x_1 \rfloor)$
$A[i] = \lfloor x_1 \rfloor$
`Izlaz:` A

```
import math
def verizni(x, n):
	A = []
	x1 = x
	A.append(math.floor(x1))
	i = 1
	while i < n and x1-math.floor(x1)>0.000001:
		x1 = 1/(x1-math.floor(x1))
		A.append(math.floor(x1))
		i += 1
	return A
```

Primjer ispisa:

```
Ulaz: verizni(5/3, 6)
Izlaz: [1, 1, 2]

Ulaz: verizni(1.41421356237, 6)
Izlaz: [1, 2, 2, 2, 2, 2]
```

3.2 Konvergente

Definicija 3.3. *Neka je* $\alpha = [a_0, a_1, \ldots]$. *Tada* $\frac{p_i}{q_i} = [a_0, a_1, \ldots, a_i]$ *nazivamo* $i-ta$ ***konvergenta broja*** α, $i \in \mathbb{N}_0$.

Ako je $\alpha = [a_0, a_1, \ldots, a_n]$, tj. $\alpha \in \mathbb{Q}$, onda se konvergente $\frac{p_i}{q_i}$ definiraju za $i = 0, \ldots, n$.

Teorem 3.4. *Neka je* $\left(\frac{p_n}{q_n}\right)$ *niz konvergenti broja* $\alpha = [a_0, a_1, \ldots]$. *Tada brojevi* p_n, q_n *zadovoljavaju rekurzije*

$$\begin{aligned} p_n &= a_n p_{n-1} + p_{n-2},\ p_0 = a_0,\ p_1 = a_0 a_1 + 1, \\ q_n &= a_n q_{n-1} + q_{n-2},\ q_0 = 1,\ q_1 = a_1, \end{aligned}$$

za $n \geq 2$.

Dokaz. Za $n = 2$ tvrdnju provjerimo direktno,

$$\frac{p_2}{q_2} = [a_0, a_1, a_2] = a_0 + \frac{1}{a_1 + \frac{1}{a_2}} = \frac{a_0 a_1 a_2 + a_0 + a_2}{a_1 a_2 + 1}.$$

Iz rekurzija i početnih uvjeta slijedi:

$$p_2 = a_2 p_1 + p_0 = a_0 a_1 a_2 + a_0 + a_2,$$

$$q_2 = a_2 q_1 + q_0 = a_1 a_2 + 1.$$

Zaključujemo da tvrdnja vrijedi za $n = 2$. Pretpostavimo da je $n > 2$ i da tvrdnja vrijedi za $n - 1$. Definirajmo brojeve p'_j, q'_j sa $\frac{p'_j}{q'_j} = [a_1, a_2, ..., a_{j+1}]$. Tada je

$$p'_{n-1} = a_n p'_{n-2} + p'_{n-3},$$

$$q'_{n-1} = a_n q'_{n-2} + q'_{n-3}.$$

No,

$$\frac{p_j}{q_j} = a_0 + \frac{1}{[a_1, ..., a_j]} = a_0 + \frac{q'_{j-1}}{p'_{j-1}} = \frac{a_0 p'_{j-1} + q'_{j-1}}{p'_{j-1}}.$$

Slijedi da je,

$$p_j = a_0 p'_{j-1} + q'_{j-1},\ q_j = p'_{j-1}.$$

Prema tome,

$$\begin{aligned} p_n &= a_0(a_n p'_{n-2} + p'_{n-3}) + (a_n q'_{n-2} + q'_{n-3}) \\ &= a_n(a_0 p'_{n-2} + q'_{n-2}) + (a_0 p'_{n-3} + q'_{n-3}) = a_n p_{n-1} + p_{n-2}, \\ q_n &= a_n p'_{n-2} + p'_{n-3} = a_n q_{n-1} + q_{n-2}. \end{aligned}$$

□

Teorem 3.5. *Za sve $n \geq -1$ vrijedi:*

$$q_n p_{n-1} - p_n q_{n-1} = (-1)^n,$$

pri čemu je $p_{-2} = 0$, $p_{-1} = 1$, $q_{-2} = 1$, $q_{-1} = 0$.

Dokaz. Tvrdnju dokazujemo primjenom principa matematičke indukcije.

Baza: $n = -1$, slijedi

$$q_{-1}p_{-2} - p_{-1}q_{-2} = 0 \cdot 0 - 1 \cdot 1 = (-1)^{-1}.$$

Pretpostavka indukcije: Pretpostavimo da tvrdnja vrijedi za $n - 1$

$$q_{n-1}p_{n-2} - p_{n-1}q_{n-2} = (-1)^{n-1}.$$

Korak: Koristeći rekurzije iz 3.2 dobivamo

$$\begin{aligned} q_n p_{n-1} - p_n q_{n-1} &= (a_n q_{n-1} + q_{n-2})p_{n-1} - (a_n p_{n-1} + p_{n-2})q_{n-1} \\ &= -(q_{n-1}p_{n-2} - p_{n-1}q_{n-2}) = -(-1)^{n-1} = (-1)^n. \end{aligned}$$

□

Algoritam 9. *Algoritam za računanje konvergenti* (u programskom jeziku Python)

```
def konvergente(L):
          p0 = L[0]
          p1 = L[0]*L[1]+1
          q0 = 1
          q1 = L[0]
          for i in range(len(L)):
                p = L[i]*p1+p0
                q = L[i]*q1+q0
```

```
            p0 = p1
            p1 = p
            q0 = q1
            q1 = q
            print(p,''/'', q)
```

Algoritam koristi Teorem 3.2, gdje su dane rekurzije za računanje p_n i q_n. Algoritmu za ulazne podatke dajemo listu u kojoj je zapisani skraćeni verižni razlomak. Primjer ispisa:

```
Ulaz: konvergente([1,1,9,2])
Izlaz: 1 / 1
       2 / 1
       19 / 10
       40 / 21
```

3.3 Diofantska linearna jednadžba

Neka su $a, b, c \in \mathbb{Z}$ i $a, b \neq 0$. Jednadžbu oblika

$$ax + by = c \tag{3.3}$$

čija rješenja tražimo u prstenu cijelih brojeva nazivamo *diofantskom linearnom jednadžbom.*

Bez smanjenja općenitosti pretpostavimo da je $\gcd(a, b) = 1$. Naime, ako je $\gcd(a, b) > 1$ i $\gcd(a, b)$ ne dijeli c, onda jednadžba (3.3) nema rješenja. Ako $\gcd(a, b)$ dijeli c, onda (3.3) podijelimo sa $d = \gcd(a, b)$ i dobivamo ekvivalentnu jednadžbu $a'x + b'y = c'$ gdje je $a' = a/d$, $b' = b/d$, $c' = c/d$ i $\gcd(a', b') = 1$.

Racionalni broj $\frac{a}{b}$ ima razvoj u verižni razlomak jednak $\frac{a}{b} = [a_0; a_1, ..., a_n]$. Kako je $\frac{a}{b} = \frac{p_n}{q_n}$, koristeći formulu iz Teorema 3.5 i uvrštavanjem $a = p_n$ i $b = q_n$ dobivamo

$$bp_{n-1} - aq_{n-1} = (-1)^n. \tag{3.4}$$

1. Ako je n neparan, tada dobivamo:

$$aq_{n-1} - bp_{n-1} = 1.$$

Neka je $(x_0, y_0) \in \mathbb{Z}^2$ neko rješenje jednadžbe (3.3). Tada je

$$ax_0 + by_0 = c \cdot 1 = c(aq_{n-1} - bp_{n-1}),$$

odnosno

$$a(cq_{n-1} - x_0) = b(y_0 + cp_{n-1}).$$

Kako je $\gcd(a, b) = 1$, slijedi da b dijeli $(cq_{n-1} - x_0)$, tj. postoji $t_0 \in \mathbb{Z}$ takav da je

$$cq_{n-1} - x_0 = bt_0,$$

odnosno

$$x_0 = cq_{n-1} - bt_0$$

te

$$y_0 = at_0 - cp_{n-1}.$$

Lako se može provjeriti da za neparan n vrijedi:

$$(x_t, y_t) = (cq_{n-1} - bt, at - cp_{n-1}) \tag{3.5}$$

je rješenje jednadžbe (3.3) za svaki $t \in \mathbb{Z}$.

2. Ako je n paran, tada dobivamo:

$$bp_{n-1} - aq_{n-1} = 1.$$

Analogno,

$$ax_0 + by_0 = c \cdot 1 = c(bp_{n-1} - aq_{n-1}),$$

odnosno

$$a(x_0 + cq_{n-1}) = b(cp_{n-1} - y_0).$$

Kako je $\gcd(a, b) = 1$, slijedi da b dijeli $(x_0 + cq_{n-1})$, tj. postoji $t_0 \in \mathbb{Z}$ takav da je

$$x_0 + cq_{n-1} = bt_0,$$

odnosno

$$x_0 = bt_0 - cq_{n-1}$$

te

$$y_0 = cp_{n-1} - at_0.$$

Za paran n vrijedi:

$$(x_t, y_t) = (bt - cq_{n-1}, cp_{n-1} - at) \tag{3.6}$$

je rješenje jednadžbe (3.3) za svaki $t \in \mathbb{Z}$.

Primjer 3.6. *Riješimo jednadžbu* $40x + 21y = 450$.

Rješenje. *Algoritam 9 nam daje konvergente verižnog razlomka* $\frac{40}{21}$.

```
Ulaz: konvergente(40,21)
Izlaz: 1 / 1
       2 / 1
       19 /  10
       40 / 21
```

Iščitajmo konvergente:

$$\frac{p_0}{q_0} = 1\,, \frac{p_1}{q_1} = 2\,, \frac{p_2}{q_2} = \frac{19}{10}.$$

Uvrštavanjem u (3.5) *dobivamo opće rješenje. Rješenje je uređeni par*

$$(x, y) = (450 \cdot 10 - 21t, 40t - 450 \cdot 19),\ t \in \mathbb{Z}.$$

Koristeći Euklidov algoritam za računanje verižnog razlomka i Algoritam 9 za računanje konvergenti, možemo riješiti diofantsku linearnu jednadžbu $ax + by = c$, gdje je $\gcd(a, b) = 1$. Algoritam za ulazne podatke prima $a, b, c \in Z$ te vraća opća rješenja jednadžbe.

Algoritam 10. *Linearna diofantska jednadžba*

```
def diofantska(a, b, c):
        L = []
        a1, b1 = a, b
        while b1!=0:              #stvaramo listu u kojoj je zapis
             L.append(a1//b1)                veriznog razlomka
             L.append(a1//b1)
             p = a1
             a1 = b1
             b1 = p%b1
        if len(L)%2==1:           #n neparan
                if gcd(a, b)==1:
                         p, q = konvergente(L)
                         print('x =', c*q,'-',b,'*t')
                         print('y =', a,'*t','-',c*p)
                else:
                         if c%gcd(a,b)!=0:
                           print('Jednadzba nema cjelobrojnih rj.')
                         else:
                           c //= gcd(a,b)
                           p, q = konvergente(L)
                           print('x =', c*q,'-',b,'*t')
                           print('y =', a,'*t','-',c*p)
        if len(L)%2==0:           #n paran
               if gcd(a, b)==1:
                       p, q = konvergente(L)
                       print('x =', b, '*t','-', -c*q)
                       print('y =', -a, '*t', '-', c*p)
               else:
                       if c%gcd(a,b)!=0:
                            print('Jednadzba nema cjelobrojnih rj.')
                       else:
                            c //= gcd(a,b)
                            p, q = konvergente(L)
                            print('x =', -b, '*t','-', -c*q)
                            print('y =', a, '*t', '-',c*p)
```

U algoritmu se koristi Algoritam 1 koji nam koristi za provjeru ima li jednadžba rješenja, to jest ako je $\gcd(a, b) = 1$. Funkcija *konvergente* vraća p_{n-1} i q_{n-1} koje koristimo u ispisu rješenja.

Primjer ispisa:

```
Ulaz: diofantska(50, 21, 450)
Izlaz: x = 21 * t + 3600
       y = -50 * t - 8550

Ulaz: diofantska(1000, 500, 2)
Izlaz: Jednadzba nema cjelobrojnih rjesenja.
```

Poglavlje 4

Ispitivanje prostosti

Testove prostosti koristimo kad želimo za zadani $n \in \mathbb{N}$ provjeriti je li broj n prost. Najprirodniji test prostosti je da provjerimo sve brojeve veće od 1 i manje od $\sqrt{n}$ jesu li djelitelji broja n. Ako dođemo do jednog djelitelja broja n zaključujemo da je n složen broj, a ako ne pronađemo ni jednog djelitelja, zaključujemo da je n prost broj. Kako prostih brojeva manjih od $\sqrt{n}$ ima $O(\sqrt{n}/\log n)$, a za svako dijeljenje treba $O(\log^2 n)$ operacija, složenost je $O(\sqrt{n}\log n)$. Složenost je prevelika da bi koristili algoritam za velike brojeve. Zbog toga postoje različiti algoritmi za testiranje prostosti koji imaju daleko bolju vremensku složenost.

4.1 Fermatov test

Prikazat ćemo vjerojatnosni test prostosti koji se zasniva na Malom Fermatovom teoremu.

Definicija 4.1. ***Reducirani sustav ostataka modulo m*** *je skup svih cijelih brojeva r_i sa svojstvom da je $\gcd(r_i, m) = 1$, $r_i \not\equiv r_j \pmod{m}$ za $i \neq j$, te za svaki cijeli broj x takav da je $\gcd(x, m) = 1$ postoji r_i takav da je $x \equiv r_i \pmod{m}$.*

Primjer jednog reducirani sustav ostataka modulo m je skup svih brojeva $a \in \{1, 2, ..., m\}$ takvih da je $\gcd(a, m) = 1$. Svaki reducirani sustav ostataka modulo m je ekvipotentan, odnosno ima isti broj elemenata.

Definicija 4.2. *Funkcija $\varphi : \mathbb{N} \to \mathbb{N}$ gdje je $\varphi(m)$ broj elemenata u reduciranom sustavu ostataka modulo m naziva se* ***Eulerova funkcija****.*

Jednostavnije, možemo reći da je $\varphi(m)$ broj brojeva u nizu $1, 2, \ldots, m$ koji su relativno prosti s m. Lako se vidi da je $\varphi(p) = p - 1$ za svaki prost broj p. Nadalje, Eulerova funkcija φ zadovoljava važno svojstvo *multiplikativnosti*. Naime, vrijedi

da je $\varphi(1) = 1$ i $\varphi(mn) = \varphi(m)\varphi(n)$ za sve $m, n \in \mathbb{N}$ koji su relativno prosti, tj. $\gcd(m, n) = 1$.

Teorem 4.3. *Neka je* $\{x_1, x_2, \ldots, x_{\varphi(m)}\}$ *potpuni sustav ostataka modulo* m*,* $a \in \mathbb{Z}$ *te* $\gcd(a, m) = 1$*. Tada je i* $\{ax_1, ax_2, \ldots, ax_{\varphi(m)}\}$ *potpuni sustav ostataka modulo* m*.*

Dokaz. Dovoljno je pokazati da je $ax_i \not\equiv ax_j \pmod{m}$ za $i \neq j$. Pretpostavimo da je $ax_i \equiv ax_j \pmod{m}$ za $i \neq j$. Tada Teorem 2.4 povlači da je $x_i \equiv x_j \pmod{m}$, to jest $i = j$. □

Teorem 4.4 (Eulerov teorem)**.** *Ako je* $\gcd(a, m) = 1$*, onda je* $a^{\varphi(m)} \equiv 1 \pmod{m}$.

Dokaz. Neka je $\{r_1, r_2, ..., r_{\varphi(m)}\}$ reducirani sustav ostataka modulo m. Tada po Teoremu 4.3 slijedi da je $\{ar_1, ar_2, ..., ar_{\varphi(m)}\}$ također reducirani sustav ostataka modulo m. Zaključujemo da je

$$\prod_{j=1}^{\varphi(m)} (ar_j) \equiv \prod_{i=1}^{\varphi(m)} r_i \pmod{m},$$

odnosno

$$a^{\varphi(m)} \prod_{j=1}^{\varphi(m)} r_j \equiv \prod_{i=1}^{\varphi(m)} r_i \pmod{m}.$$

Kako je $\gcd(r_i, m) = 1$, primjenom Teorema 2.4, dobivamo $a^{\varphi(m)} \equiv 1 \pmod{m}$. □

Korolar 4.5 (Mali Fermatov teorem)**.** *Neka je* p *prost broj. Ako* $p \nmid a$*, onda je* $a^{p-1} \equiv 1 \pmod{p}$*. Za svaki cijeli broj* a *vrijedi* $a^p \equiv a \pmod{p}$.

Dokaz. Očito je $\varphi(p) = p - 1$, pa tvrdnja slijedi iz Teorema 4.4. □

Prethodnu tvrdnju možemo koristiti za testiranje prostosti. Naime, ako je n složen, onda $a^{n-1} \not\equiv 1 \pmod{n}$. Stoga smo sigurni da ako n ne zadovoljava Mali Fermatov teorem, onda nije prost. No, ako ga zadovoljava, tj. ako je $a^{n-1} \equiv 1 \pmod{n}$ onda je n možda ili vjerojatno prost. Vjerojatnost je veća ako smo test proveli za više različitih baza a. Međutim, iako je za veliku većinu brojeva dovoljno svega nekoliko puta ponoviti Fermatov test da bi se pokazala njihova složenost, postoje i brojevi za koje test ne daje točan odgovor. To su tzv. *Carmichaelovi brojevi*. Složen broj n je Carmichaelov ako je $a^{n-1} \equiv 1 \pmod{n}$ za svaki a koji je relativno prost s n. Najmanji Carmichaelov broj je $561 = 3 \cdot 11 \cdot 17$. Srećom, Carmichaelovi brojevi su vrlo rijetki, no ipak se sluti da ih je beskonačno mnogo.

Algoritam 11. *Frematov test prostosti I*

`Ulaz:` $n, k \in \mathbb{N}$
$k = 1$
`ponovi` za $i \leq k$ `puta`
`odaberi` $a \in \{2, ..., n-1\}$
`ako je` $a^{n-1} \neq 1$ `onda` $i = i + 1$
`inače Izlaz:` n `nije prost`
$i = k + 2$
`ako je` $i = k + 1$ onda `Izlaz:` n `vjerojatno prost`

Fermatov test prostosti u programskom jeziku Python:

```
import random

def fermat_test(n, k):
          if n == 2:                #2 je prost broj
                return True
          if n%2 == 0:              #parni brojevi su složeni
                return True
          for i in range(k):
                a = random.randint(1,n-1)      #slučajni odabir
                if pow(a, n-1) % n !=1:
                        return False
          return True
```

Pomoću sljedeće tvrdnje Fermatov test može se "pojačati".

Propozicija 4.6. *Ako je p prost i x zadovoljava kongruenciju $x^2 \equiv 1 \pmod{p}$, tada je $x \equiv 1 \pmod{p}$ ili $x \equiv -1 \pmod{p}$.*

Dokaz. Ako je $x^2 \equiv 1 \pmod{p}, x \neq 0$, slijedi da je $x^2 - 1 = kp, k \in \mathbb{Z}$. $x^2 - 1 = (x-1)(x+1)$ te zaključujemo da $p \mid (x-1)(x+1)$. Iz toga slijedi da p dijeli $(x-1)$ ili p dijeli $(x+1)$. □

Uočimo da za p prost (neparan) i $a \in \mathbb{N}$ takav da $p \nmid a$ vrijedi da je

$$\left(a^{(p-1)/2}\right)^2 \equiv 1 \pmod{p}.$$

Algoritam 12. *Fermatov test prostosti II*

Ulaz: $n \in \mathbb{N}$
odaberi $a \in \{2, \ldots, n-1\}$
ako je $a^{(n-1)/2} \mod n \neq 1$ ili $a^{(n-1)/2} \mod n \neq -1$ onda
Izlaz: n nije prost
inače
Izlaz: n vjerojatno prost

Ovaj test prostosti, osim što je nešto jednostavniji zbog potenciranja broja a, računamo $a^{(n-1)/2}$ umjesto a^{n-1}, već je i jači. Na primjer, $3^{6601} \equiv 1 \pmod{6601}$ pa bi prema Algoritmu 11 slijedilo da je možda prost, dok zbog $3^{3300} \equiv 4509 \pmod{6601}$ slijedi da je 6601 složen broj prema Algoritmu 12.

Algoritam u programskom jeziku Python:

```
import random
def test_prost(n):
          a = random.randint(2, n-1)
          m = int((n-1)/2)
          x = pow(a, m) % n
          if x == 1 or x == n-1:
               return True
          else:
               return False
```

Primjer ispisa:

```
Ulaz: test_prost(4031)
Izlaz: False
Ulaz: test_prost(4799)
Izlaz:True
```

4.2 Millerov test

Ideju koju smo koristili za Algoritam 12 možemo još razraditi. Pretpostavimo da je $a^{(n-1)/2} \equiv 1 \pmod{n}$ i $\frac{n-1}{2}$ je paran. Ako je n prost tada slijedi

$$a^{(n-1)/4} \equiv \pm 1 \pmod{n}.$$

Nadalje, ako je $a^{(n-1)/4} \equiv 1 \pmod{n}$ i $\frac{n-1}{4}$ paran tada n prost povlači da je

$$a^{(n-1)/8} \equiv \pm 1 \pmod{n},$$

itd. Na primjer, za $n = 6601$ i bazu $a = 2$ slijedi $2^{3300} \equiv 1 \pmod{6601}$. Nadalje, iz

$$2^{1650} \equiv 4509 \pmod{6601}$$

zaključujemo da je 6601 složen broj. Algoritam koji slijedi iz opisanog postupka naziva se Millerov test prostosti u bazi a.

Algoritam 13. *Millerov test prostosti u bazi a*

```
Ulaz:  n, a ∈ ℕ, 1 < a < n, n neparan
t = 1
e = (n − 1)/2
Sve dok t = 1
    a^e ≡ u (mod n)
    Ako u mod = −1 onda
      t = 0
    inače
      Ako u mod = 1 onda
            Ako  e neparan onda
              e = e/2
            inače
              t = 2
Ako je t = 2 onda
      Izlaz: n nije prost
inače
      Izlaz: n je prošao Millerov test u bazi a
```

```
def miller(n, a):
          test = 0
          e = int((n-1)/2)
          while pow(a, e)%n == 1 or pow(a, e)%n == n-1:
               if n % 2 == 1:
                    e = int(e/2)
               test = 1
          return test
```

Procedura vraća 1 ako je broj prošao Millerov test u danoj bazi, a 0 ako je broj n složen. Primjer ispisa:

```
Ulaz: miller(7906)
Izlaz: 1
Ulaz: miller(90193)
Izlaz: 0
```

Da bi broj n prošao Millerov test, u algoritmu je zadovoljen jedan od uvjeta:

1. $a^{(n-1)/2^i} \equiv -1 \pmod{n}$, za $i > 0$, gdje $2^i \mid n-1$
2. $a^{(n-1)/2^i} \equiv 1 \pmod{n}$, gdje je i najveća potencija od 2 koja dijeli $n-1$.

U suprotnom, postoji paran broj m za koji vrijedi

$$a^m \equiv 1 \pmod{n},\ a^{m/2} \not\equiv \pm 1 \pmod{n},$$

a to povlači da je n složen broj.

Zanimljivo je da je $3215031751 = 151 \cdot 751 \cdot 28351$ jedini složeni broj manji od $2.5 \cdot 10^{10}$ koji prolazi Millerov test za baze $2, 3, 5$ i 7, ali u bazi 11 ne prolazi test, odnosno pokazuje da je složen.

Poglavlje 5

RSA kriptosustav

5.1 Definicija kriptosustava

Kriptografija, ili u prijevodu *tajnopis*, proučava metode za slanje poruka koje trebaju biti čitljive samo onima kojima su namijenjene. Pošiljatelj primatelju šalje poruku, tzv. *otvoreni tekst*, preko nesigurnog komunikacijskog kanala (radio veza, internet, telefonska linija) i stoga ju mora transformirati, odnosno *šifrirati*, kako protivnici ne bi otkrili njen sadržaj. Otvoreni tekst se pomoću unaprijed dogovorenog *ključa* zamjenjuje *šifratom*. Primatelj, koji jedini posjeduje ključ za dešifriranje, šifrat može transformirati u otvoreni tekst, tj. otvorenu poruku. Ideja klasičnog kriptosustava bazira se na tome da pošiljatelj i primatelj moraju razmijeniti ključeve za šifriranje, a ključ za dešifriranje se lako dobiva iz ključa za šifriranje, ili je čak isti.

Definicija 5.1. *Kriptosustav je uređena petorka* $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$*, gdje su:*

- $\mathcal{P}$ *konačan skup svih mogućih osnovnih elemenata otvorenog teksta;*
- $\mathcal{C}$ *konačan skup svih mogućih osnovnih elemenata šifrata;*
- $\mathcal{K}$ *konačan skup svih mogućih ključeva;*
- $\mathcal{E}$ *skup svih funkcija šifriranja;*
- $\mathcal{D}$ *skup svih funkcija dešifriranja;*

pri čemu za svaki $K \in \mathcal{K}$ *postoji funkcija šifriranja* $e_K \in \mathcal{E}$ *i odgovarajuća funkcija dešifriranja* $d_K \in \mathcal{D}$*. Nadalje, funkcije* $e_K : \mathcal{P} \to \mathcal{C}$ *i* $d_K : \mathcal{C} \to \mathcal{P}$ *imaju svojstvo da je* $d_K(e_K(x)) = x$*, za svaki* $x \in \mathcal{P}$*.*

Postoje različite vrste podjela kriptosustava. Jedna od njih je podjela prema javnosti, odnosno tajnosti ključa. Ako se ključ za dešifriranje može izračunati poznavajući ključ za šifriranje i obratno, ili su ključevi identični, onda govorimo o *kriptosustavima s tajnim ključem* ili *simetričnim kriptosustavima*. Jasno je da njihova sigurnost leži u tajnosti ključa. Kriptosustavi kojima se ključ za dešifriranje ne može lako izračunati iz poznavanja ključa za šifriranje nazivaju se *kriptosustavi s javnim ključem* ili *asimetrični kriptosustavi*. Ovi sustavi zasnovani su na ideji da bilo tko može šifrirati poruku pomoću *javnog* ključa, ali za dešifriranje je potreban *tajni* ili *privatni* ključ.

RSA kriptosustav je najpoznatiji kriptosustav s javnim ključem. Osmislili su ga Ronald Rivest, Ad Shamir i Leonard Adleman 1978. godine.

Definicija 5.2 (RSA kriptosustav)**.** *Neka je $n = pq$, gdje su p i q prosti brojevi. Neka je $\mathcal{P} = \mathcal{C} = \mathbb{Z}_n$ te*

$$\mathcal{K} = \{(n, p, q, d, e) : n = pq,\ de \equiv 1 \pmod{\varphi(n)}\}.$$

Tada za $k \in \mathcal{K}$ definiramo

$$e_k(x) = x^e \mod n,\ d_k(y) = y^d \mod n,\ x, y \in \mathbb{Z}_n.$$

Vrijednosti n i e su javne, a vrijednosti p, q i d su tajne. Kažemo (n, e) je javni ključ, a (p, q, d) je tajni ključ.

Uočimo da je $\varphi(n) = \varphi(p \cdot q) = \varphi(p) \cdot \varphi(q) = (p-1) \cdot (q-1) = pq - p - q + 1 = n - p - q + 1$.

5.2 Generiranje ključeva i šifriranje

Algoritam 14. *Algoritam za generiranje ključeva u RSA kriptosustavu*

1. Slučajnim odabirom odabiru se dva različita broja p i q tako da svaki od njih ima barem 100 znamenaka. (Nezamislivo je nasumično odabrati tako veliki prost broj. Stoga se najprije odabere prirodan stoznamenkasti broj te pomoću testova prostosti tražimo prve proste brojeve koji su veći od odabranog.)

2. Računa se $n = pq$ i $\varphi(n) = (p-1)(q-1)$.

3. Odabiremo e takav da je $\gcd(e, \varphi(n)) = 1$. Često se za eksponent e biraju Fermatovi prosti brojevi 17 i 65537. (Njihovim odabirom ubrzava se proces modularnog potenciranja koji se koristi kod šifriranja i dešifriranja. Broj 3 se

ne preporuča jer se tada slučaju do otvorenog teksta može doći bez poznavanja ključa za dešifriranje, rješavanjem odgovarajućeg sustava linearnih kongruencija.)

4. Računa se d kao modularni inverz od e (modulo $\varphi(n)$), odnosno rješava se linearna kongruencija
$$de \equiv 1 \pmod{\varphi(n)}.$$
(d se može računati pomoću Algoritma 4.)

5. Uređeni par (n, e) objavljuje se kao javni, a (p, q, d) je tajni ključ.

U sljedećem primjeru prikazujemo kako funkcionira RSA kriptosustav s malim parametrima.

Primjer 5.3. *Zadani su prosti brojevi $p = 7$ i $q = 11$. Za prikladno odabran e šifrirajte poruku $x = 24$. Nadalje, odredite tajni ključ koji odgovara odabranom e.*

Rješenje. Imamo da je $n = 77$ i $\varphi(n) = \varphi(77) = \varphi(7) \cdot \varphi(11) = 60$. Odaberimo neki e koji je relativno prost s 60, na primjer $e = 13$. Dakle, javni ključ je $(77, 13)$. Šifrat glasi
$$y = e(24) = 24^{13} \mod 77 = 52.$$
Da bi primatelj mogao izračunati otvoreni tekst potreban mu je tajni ključ kojeg može odrediti pomoću *skrivenog podatka* - faktorizacije od n, odnosno prostih brojeva p i q. Tajni ključ određujemo rješavanjem linearne kongruencije
$$ed \equiv 1 \pmod{\varphi(n)}.$$
Ukoliko ne bismo znali faktorizaciju broja n tajni ključ bi bilo gotovo nemoguće odrediti za dovoljno veliki n. U našem slučaju je sada
$$13d \equiv 1 \pmod{60}$$
i $d = 37$. Dakle, tajni ključ je $(7, 11, 37)$. Za provjeru dešifrirajmo šifrat:
$$d(52) = 52^{37} \mod 77 = 24 = x.$$

Za efikasan algoritam RSA kriptosustava, važno je da se modularno potenciranje izvodi brzo. Osnovna metoda za računanje izraza $x^e \mod n$ naziva se *kvadriraj i množi*. Prikažimo najprije e u bazi 2:
$$e = 2^{s-1} \cdot e_{s-1} + \cdots + 2 \cdot e_1 + e_0,$$

gdje je $e_{s-1} = 1$ i $e_0, e_1, \ldots, e_{s-2} \in \{0, 1\}$. Zapišimo eksponent e kao

$$e = ((\cdots((e_{s-1} \cdot 2 + e_{s-2}) \cdot 2 + e_{s-3}) \cdot 2 + \cdots + e_2) \cdot 2 + e_1) \cdot 2 + e_0.$$

Sada x^e možemo zapisati kao

$$x^e = \left(\left(\cdots\left(\left(x^{2e_{s-1}} \cdot x^{e_{s-2}}\right)^2 \cdot x^{e_{s-3}}\right)^2 \cdots x^{e_2}\right)^2 \cdot x^{e_1}\right)^2 \cdot x^{e_0}$$

iz čega slijedi sljedeći algoritam:

Algoritam 15. *Kvadriraj i množi*

`Ulaz:` x
  $y = 1$
  `za` $i = s-1, \ldots, 1, 0$ `radi`
    $y = y^2 \mod n$
    `ako je` $e_i = 1$ `onda` $y = y \cdot x \mod n$
`Izaz:` x

Bibliografija

[1] E. Bach, J. Shallit, *Algorithmic Number Theory, Volume I: Efficient Algorithms*, The MIT Press, 1996.

[2] H. M. Edwards, *Higher arithmetic: an algorithmic introduction to number theory*, AMS, 2008.

[3] J. Buhler, S. Wagon, *Basic algorithms in number theory*, Algorithmic Number Theory MSRI Publications, Volume 44, 2008.

[4] N. P. Smart, *Cryptography Made Simple*, Springer, 2016.

[5] *Geeks for Geeks*, `https://www.geeksforgeeks.org` (20.6.2019.)

[6] *Python*, `https://www.python.org/` (20.6.2019.)

Sažetak

Algoritam je metoda za rješavanje nekog problema. U ovom radu su opisani algoritmi teorije brojeva. Opisuju se Euklidov algoritam, algoritam temeljen na Kineskom teoremu o ostatcima, algoritmi za modularne operacije i algoritam za razvoj broja u verižni razlomak. Navedeni algoritmi koriste se za ispitivanje prostih brojeva, rješavanje nekih diofantskih jednadžbi i za RSA kriptosustav s javnim ključem. Svi prezentirani algoritmi zapisani su u programskom jeziku Python 3.6.

Summary

An algorithm is a step by step method of solving a problem. In this thesis some standard algorithms of number theory are described. We describe Euclidean algorithm, the algorithm based on Chinese remainder theorem, modular arithmetic algorithms and continued fraction algorithms. We show how these algorithms can be applied for prime numbers testing, solving Diophantine equations and for RSA public key cryptography. All presented algorithms are written in a programming language Python 3.6.

Životopis

Petra Međimurec rođena je 29.4.1994. u Čakovcu. Živi u Donjoj Dubravi gdje je polazila Osnovnu školu Donja Dubrava od 2001. do 2009. godine. Srednjoškolsko obrazovanje nastavila je u Gimaziji Fran Galović u Koprivnici gdje je upisala prirodoslovno – matematički smjer. Godine 2013. završila je srednju školu te položila državnu maturu. Iste je godine upisala preddiplomski sveučilišni studij Matematika; smjer: nastavnički, na Prirodoslovno - matematičkom fakultetu u Zagrebu. Preddiplomski studij je završila 2017. godine kada je upisala diplomski sveučilišni studij Matematika i informatika; smjer: nastavnički na istom fakultetu.